Parallel Debugging Techniques

Le Yan
Louisiana Optical Network Initiative

CENTER FOR COMPUTATION
& TECHNOLOGY

Outline

- Overview of parallel debugging
  - Challenges
  - Tools
  - Strategies
- Get familiar with TotalView/DDT through hands-on exercises

CENTER FOR COMPUTATION & TECHNOLOGY

Bugs in Parallel Programming

- Parallel programs are prone to the usual bugs found in sequential programs
  - Improper pointer usage
  - Stepping over array bounds
  - Infinite loops
  - …
- Plus…

CENTER FOR COMPUTATION & TECHNOLOGY

Common Types of Bugs in Parallel Programming

- Erroneous use of language features
  - Mismatched parameters, missing mandatory calls etc.
- Defective space decomposition
- Incorrect/improper synchronization
- Hidden serialization
- ……

CENTER FOR COMPUTATION & TECHNOLOGY

http://www.hpcbugbase.org/index.php/Main_Page

Debugging Essentials

- Reproducibility
  - Find the scenario where the error is reproducible
- Reduction
  - Reduce the problem to its essence
- Deduction
  - Form hypotheses on what the problem might be
- Experimentation
  - Filter out invalid hypotheses

CENTER FOR COMPUTATION & TECHNOLOGY

Terrence Parr, *Learn The Essentials of Debugging*
http://www.ibm.com/developerworks/web/library/wa-debug.html?ca=dgr-lnxw03Dbug

Challenges in Parallel Debugging

- Reproducibility
  - Many problems cannot be easily reproduced
- Reduction
  - Smallest scale might still be too large and complex to handle
- Deduction
  - Need to consider concurrent and interdependent program instances
- Experimentation
  - Cyclic debugging might be very expensive

CENTER FOR COMPUTATION & TECHNOLOGY

A Nasty Little Bug

```
…
integer*4 :: i,ista,iend
integer*4 :: chunksize=1024*1024
…
call MPI_Comm_Rank(MPI_COMM_WORLD, &
   myrank,error)
…
ista=myrank*chunksize+1
iend=(myrank+1)*chunksize
do i = ista,iend
   …
enddo
…
```

- What is the potential problem?

A Nasty Little Bug

```
…
integer*4 :: i,ista,iend
integer*4 :: chunksize=1024*1024
…
call MPI_Comm_Rank(MPI_COMM_WORLD, &
   myrank,error)
…
ista=myrank*chunksize+1
iend=(myrank+1)*chunksize
do i = ista,iend
   …
enddo
…
```

Integer overflow if myrank ≥ 4096 !

- A bug that shows up only when running with more than 4096 cores

printf/write Debugging

- Extremely easy to use, therefore dangerously attractive, but…
  - Need to edit, recompile and rerun when additional information is desired
  - May change program behavior
  - Only capable of displaying a subset of the program's state
  - Output size grows rapidly with increasing core count and harder to comprehend
- Not scalable, not recommended

CENTER FOR COMPUTATION & TECHNOLOGY

Compilers Can Help

- Most compilers can (at runtime)
  - Check array bounds
  - Trap floating operation errors
  - Provide traceback information
- Relatively scalable, but…
  - Overhead added
  - Limited capability
  - Non-interactive

Parallel Debuggers

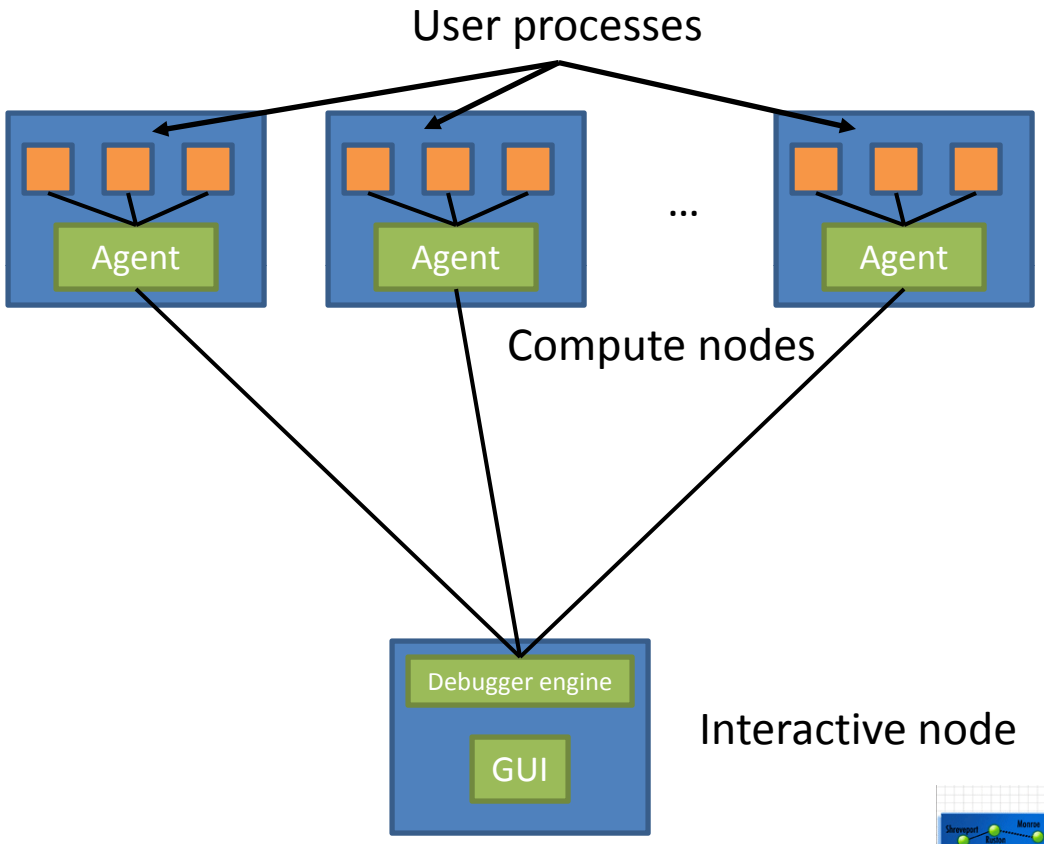
- Capable of what serials debuggers can do
  - Control program execution
  - Set action points
  - View/change values of variables
- More importantly
  - Control program execution at various levels
    - Group/process/thread
  - View MPI message queues

An Ideal Parallel Debugger

- Should allow easy process/thread control and navigation
- Should support multiple high performance computing platforms
- Should not limit the number of processes being debugged and should allow it to vary at runtime

How Parallel Debuggers Work

- Frontend
  - GUI
  - Debugger engine
- Debugger Agents
  - Control application processes
  - Send data back to the debugger engine to analyze



At Very Large Scale

- The debugger itself becomes a large parallel application
- Bottlenecks
  - Debugger framework startup cost
  - Communication between frontend and agents
  - Access to shared resources, e.g. file system

CENTER FOR COMPUTATION
& TECHNOLOGY

Validation Is Crucial

- Have a solid validation procedure to check the correctness
- Test smaller components before putting them together

General Parallel Debugging Strategy

- Incremental debugging
  - Downscale if possible
    - Participating processes, problem size and/or number of iterations
    - Example: run with one single thread to detect scope errors in OpenMP programs
  - Add more instances to reveal other issues
    - Example: run MPI programs on more than one node to detect problems introduced by network delays

CENTER FOR COMPUTATION & TECHNOLOGY

Strategy at Large Scale

- Again, downscale if possible
- Reduce the number of processes to which the debugger is attached
  - Reduces overhead
  - Reduces the required number of license seats as well
- Focus on one or a small number of processes/threads
  - Analyze call path and message queues to find problematic processes
  - Control the execution of as few processes/threads as possible while keeping others running
    - Provides the context where the error occurs

Trends in Debugging Technology

- Lightweight trace analysis tools
  - Help to identify processes/threads that have similar behavior and reduce the search space
  - Complementary to full feature debuggers
  - Example: Stack Trace Analysis Tool (STAT)
- Replay/Reverse execution
  - ReplayEngine now available from TotalView
  - Checkpointing supported in DDT 2.4
- Post-mortem statistical analysis
  - Detect anomalies by analyzing profile dissimilarity of multiple runs

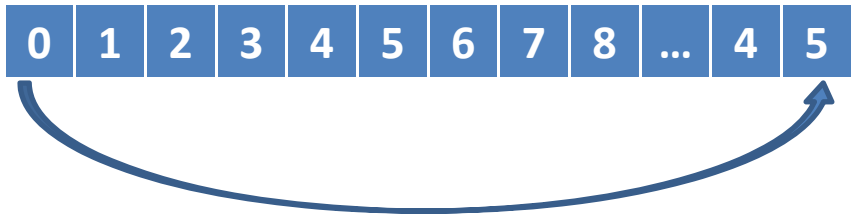
CENTER FOR COMPUTATION & TECHNOLOGY

Hands-on Exercise

- Debug MPI and OpenMP programs that solve a simple problem to get familiar with
  - Basic functionalities of parallel debuggers
    - TotalView: Pople, Bluefire and Athena
    - DDT: Ranger
  - Some common types of bugs in parallel programming
- Programs and instructions can be found at http://www.cct.lsu.edu/~lyan1/summerschool09

CENTER FOR COMPUTATION & TECHNOLOGY

Problem



- A 1-D periodic array with *N* elements
- Initial value
  - C: *cell(x)=x%10*
  - Fortran: *cell(x)=mod(x-1,10)*
- In each iteration, all elements are updated with the value of two adjacent elements:
  - $cell(x)_{i+1}=[cell(x-1)_i+cell(x+1)_i]\%10$
- Execute N_{iter} iterations
- The final outputs are the global maximum and average

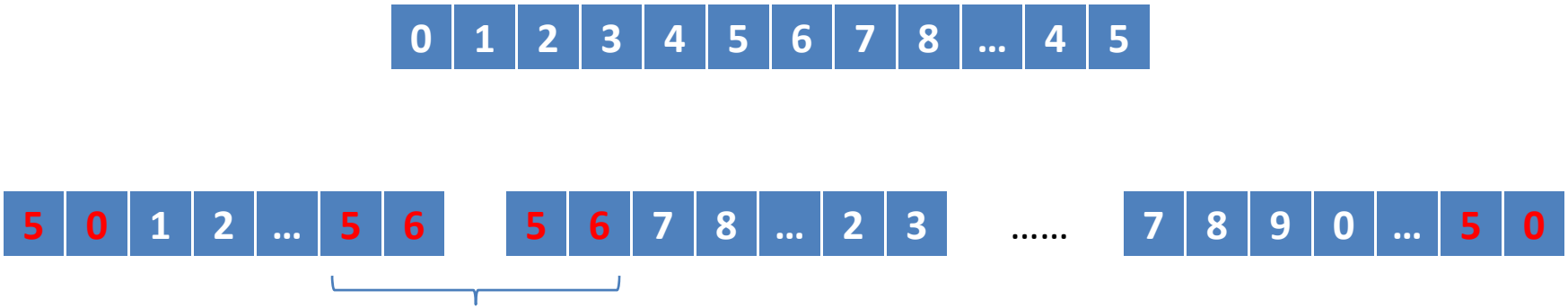
CENTER FOR COMPUTATION & TECHNOLOGY

http://www.hpcbugbase.org/index.php/Main_Page

Sequential Program

- Use an integer array to hold current values
- Use another integer array to hold the calculated values
- Swap the pointers at the end of each iteration
- The result is used to check the correctness of the parallel programs
  - Chances are that we will not have such a luxury for large jobs

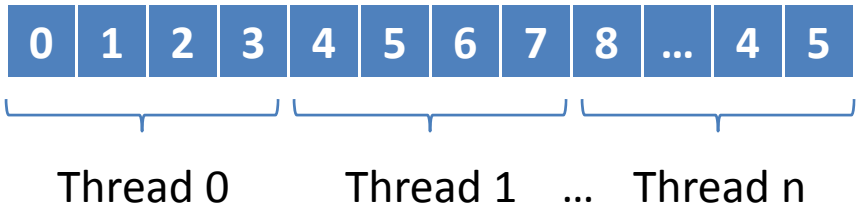
MPI Program



- Divide the array among *n* processes
- Each process works on its local array
- Exchange boundary data with neighbor processes at the end of each iteration
- Ring topology

CENTER FOR COMPUTATION & TECHNOLOGY

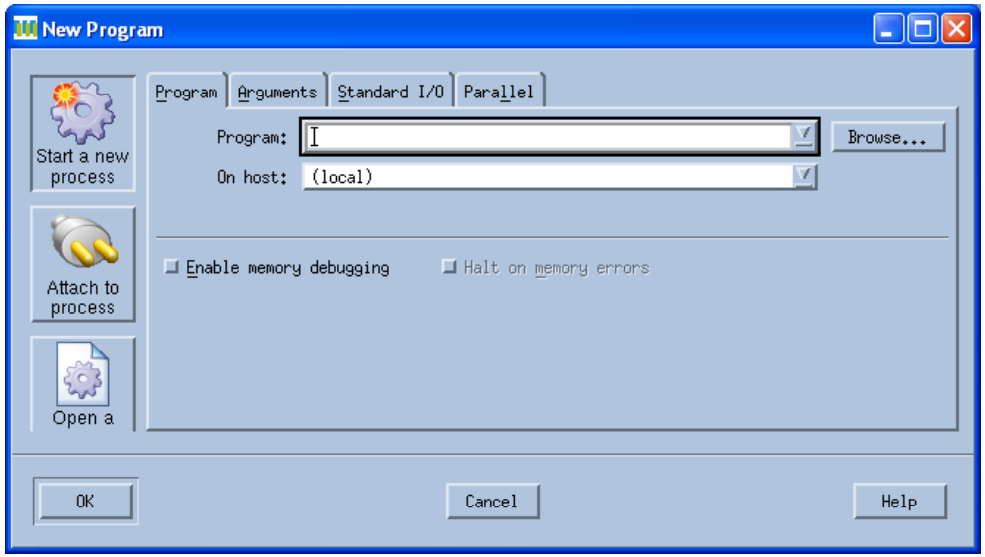
OpenMP Program



- Each thread works on its own part of the global array
- All threads have access to the entire array, so no data exchange is necessary

CENTER FOR COMPUTATION & TECHNOLOGY

Three Ways to Start TotalView/DDT

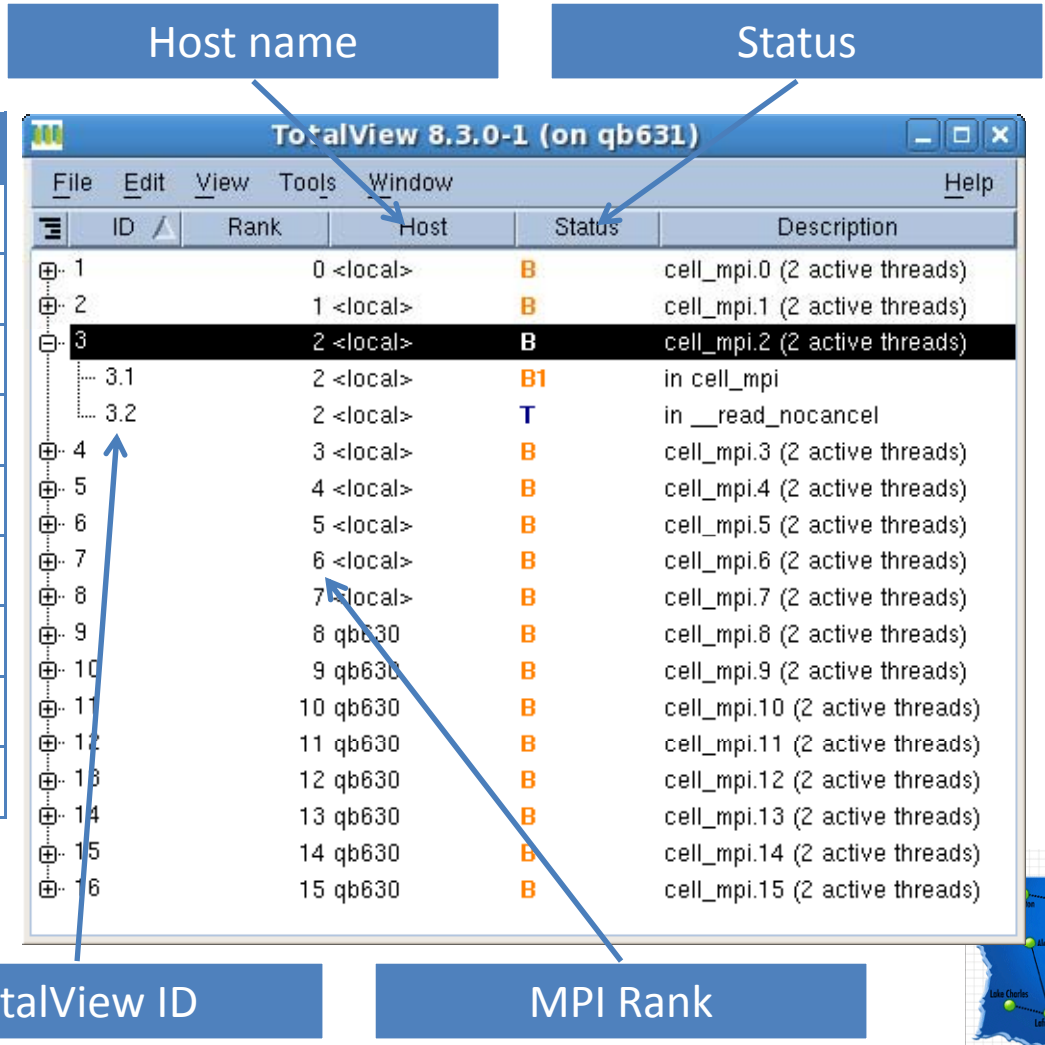


- Start with core dumps
- Start by attaching to one or more running processes
- Start the executable within TotalView/DDT

CENTER FOR COMPUTATION & TECHNOLOGY

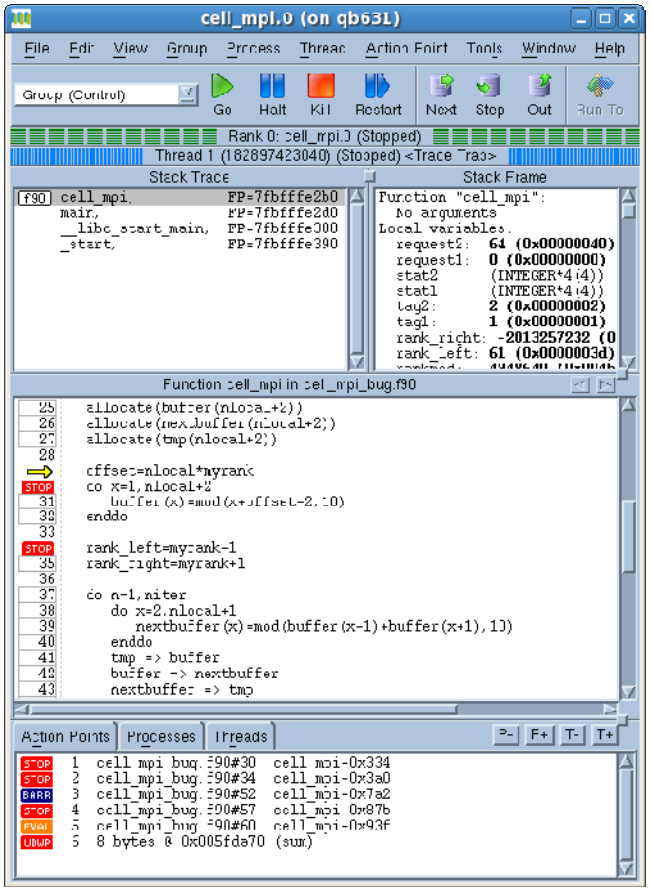
TotalView – Root Window

Status Code	Description
Blank	Exited
B	At breakpoint
E	Error
H	Held
K	In kernel
M	Mixed
R	Running
T	Stopped
W	At watchpoint

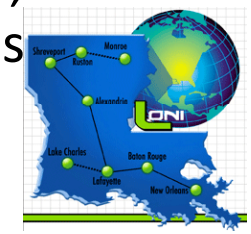


LSU
CENTER FOR COMPUTATION & TECHNOLOGY

TotalView – Process Window

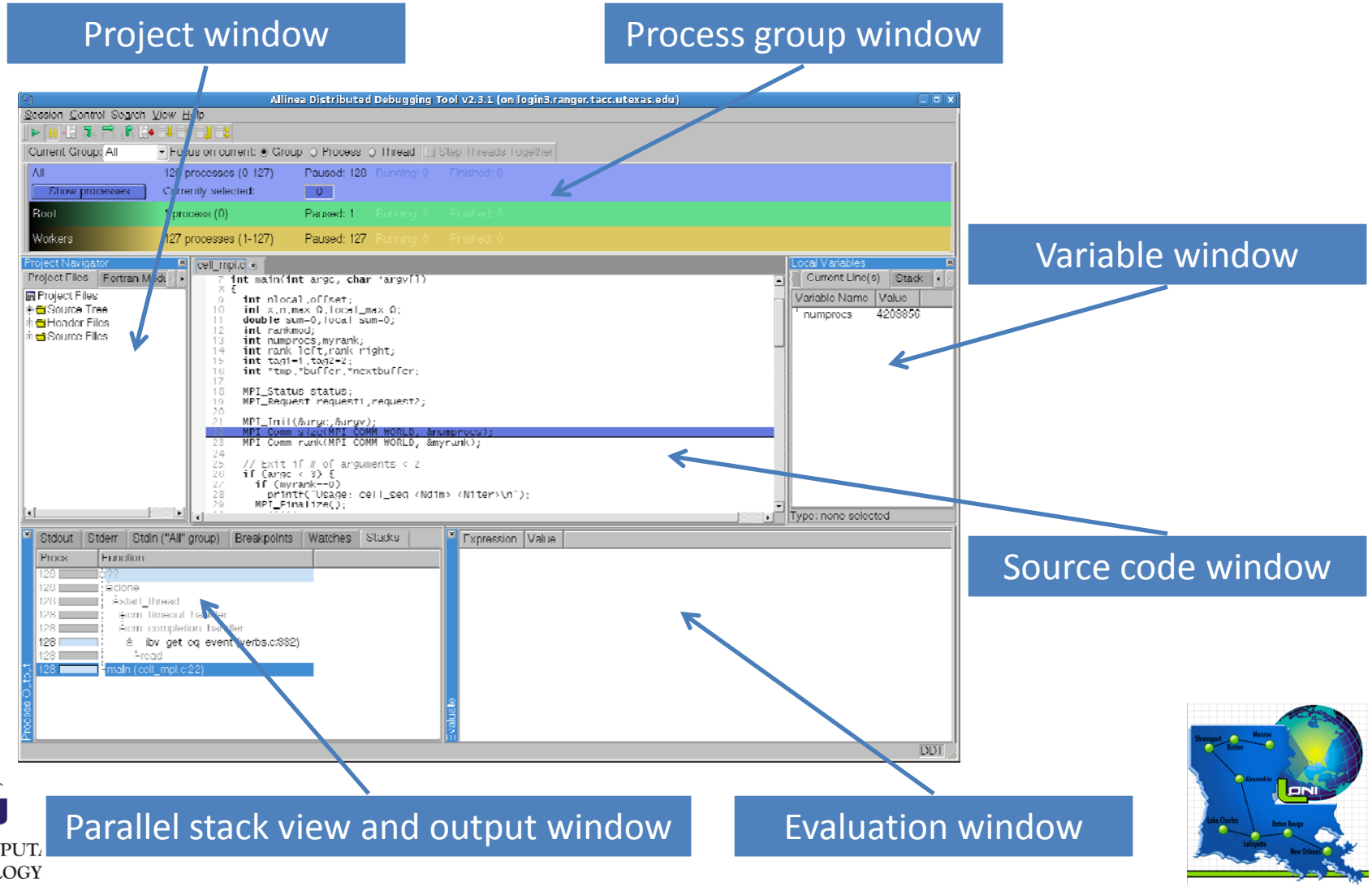


- Stack trace pane
  - Call stack of routines
- Stack frame pane
  - Local variables, registers and function parameters
- Source pane
  - Source code
- Action points, processes, threads pane
  - Manage action points, processes and threads



LSU
CENTER FOR COMPUTATION & TECHNOLOGY

DDT - Main Window



Controlling Execution

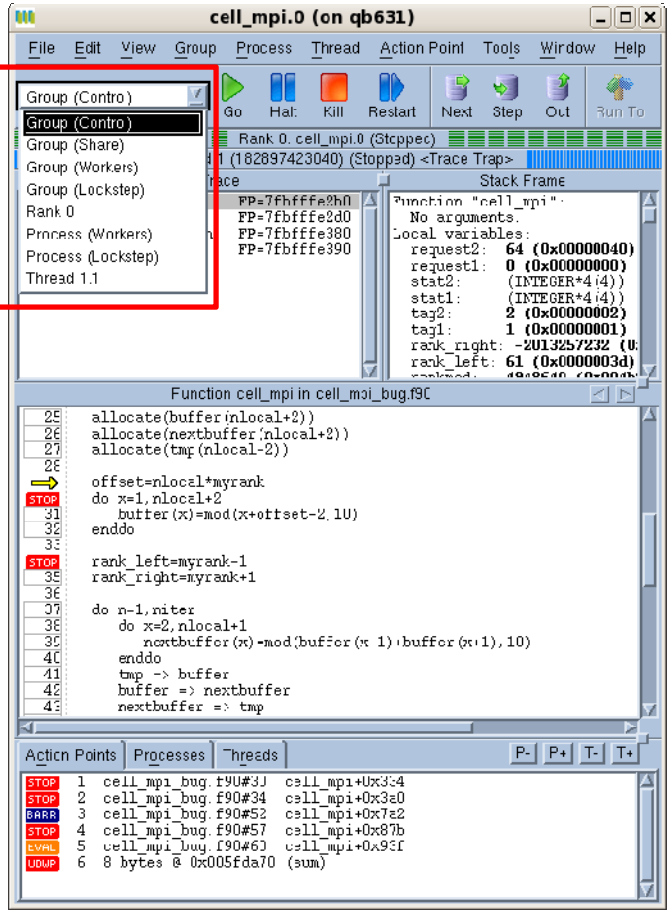
- The process window (TotalView) or main window (DDT) always focuses on one process/thread
- Switch between processes/threads
  - TotalView: p+/p-, t+/t-, double click in root window, process/thread tab
  - DDT: click on process rank in process window
- Need to set the appropriate scope when
  - Giving control commands
  - Setting action points

CENTER FOR COMPUTATION & TECHNOLOGY

Control Commands

TotalView	DDT	Description
Go	Play/Continue	Start/resume execution
Halt	Pause	Stop execution
Kill		Terminate the job
Restart		Restarts a running program
Next	Step over	Run to the next source line without stepping into another function
Step	Step into	Run to next source line
Out	Step out	Run to the completion of current function
Run to	Run to line	Run to the indicated location

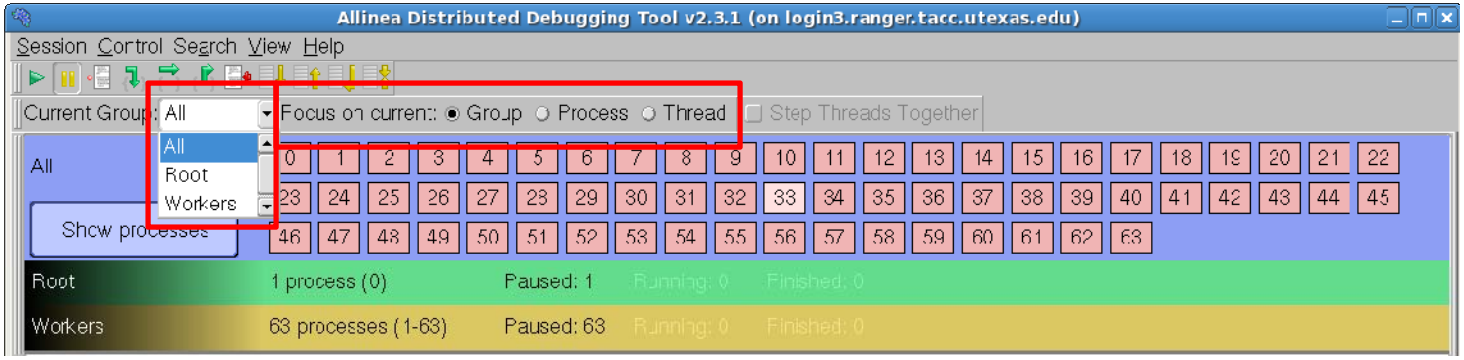
Process/Thread Groups - TotalView



- Scope of commands and action points
  - Group(control)
    - All processes and threads
  - Group(workers)
    - All threads that are executing user code
  - Rank X
    - Current process and its threads
  - Process(workers)
    - User threads in the current process
  - Thread X.Y
    - Current thread
  - User defined group
    - Group -> Custom Groups, or
    - Create in call graph

LSU
CENTER FOR COMPUTATION & TECHNOLOGY

Process/Thread Groups - DDT



- Create custom groups
  - Ctrl+click on all desired processes
  - Right click on the process window then "create group"

CENTER FOR COMPUTATION & TECHNOLOGY

Action Points

- **Breakpoint**s stop the execution of the processes and threads that reach it
  - Unconditional
  - Conditional: stop only if the condition is satisfied
  - Evaluation: stop and execute a code fragment when reached
    - Useful when testing small patches
- **Process barrier point**s synchronize a set of processes or threads
  - TotalView only
- **Watchpoint**s monitor a location in memory and stop execution when its value changes

CENTER FOR COMPUTATION & TECHNOLOGY

Setting Action Points - TotalView

- Breakpoints
  - Right click on a source line -> Set breakpoint
  - Click on the line number
- Watch points
  - Right click on a variable -> Create watchpoint
- Barrier points
  - Right click on a source line -> Set barrier
- Edit action point property
  - Right click on a action point in the Action Points tab -> Properties

CENTER FOR COMPUTATION & TECHNOLOGY

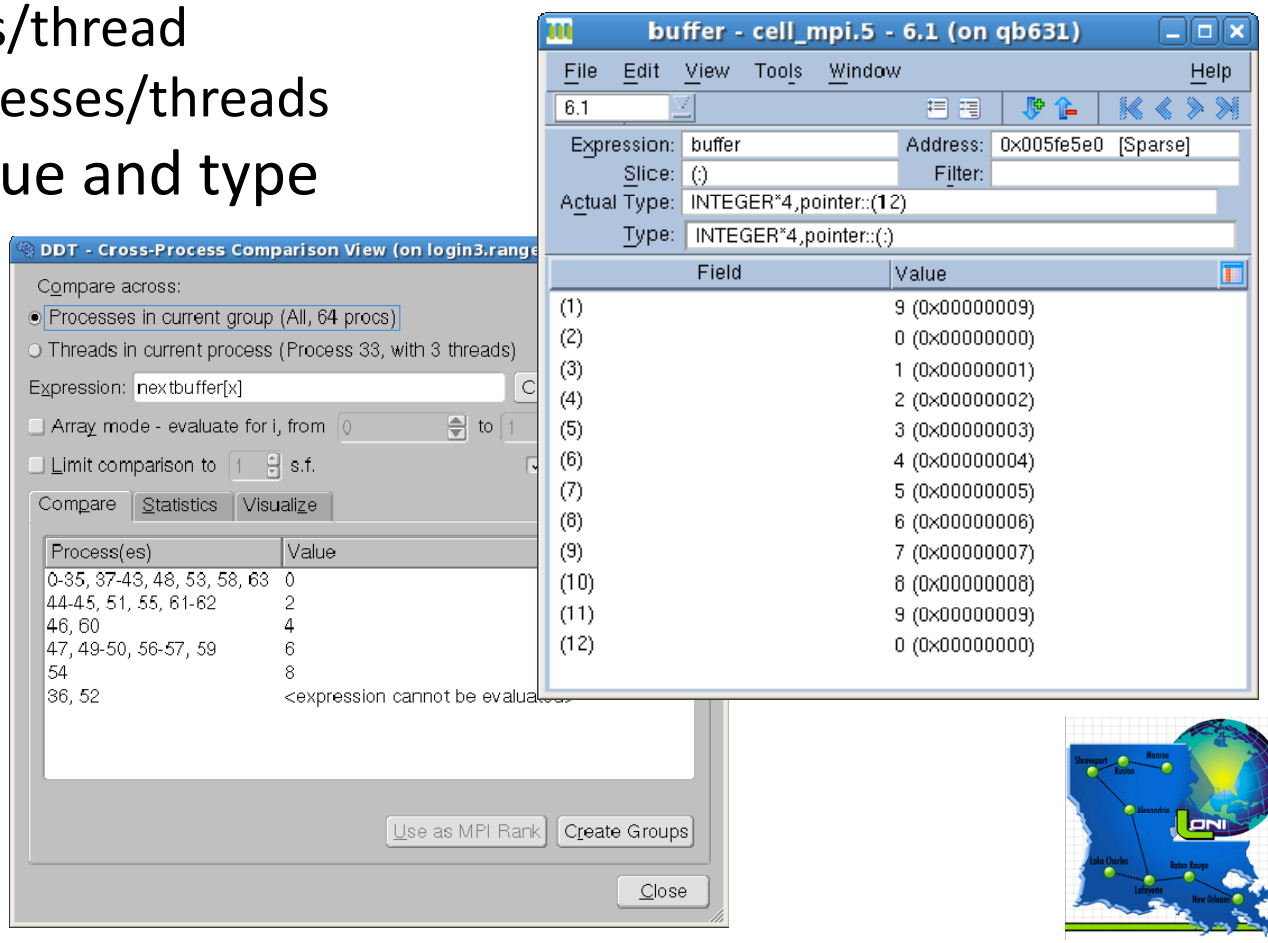
Setting Action Points - DDT

- Breakpoints
  - Double click on a source code line
  - Right click in the Breakpoints tab -> Add breakpoint
- Watch points
  - Right click on a variable -> Add to Watches
  - Right click in the Watches tab -> Add Watch

CENTER FOR COMPUTATION & TECHNOLOGY

Viewing/Editing Data

- View values and types of variables
  - At one process/thread
  - Across all processes/threads
- Edit variable value and type
- Array Data
  - Slicing
  - Filtering
  - Visualization
  - Statistics



CENTER FOR COMPUTATION & TECHNOLOGY

Viewing/Editing Data - TotalView

- Viewing data in
  - Stack frame
  - Expression list
  - Variable window (dive on a variable by double clicking on its name)
- Editing data by clicking on the value in
  - Stack frame
  - Variable window

Viewing/Editing Data - DDT

- Viewing data in
  - Variable window (in the main window)
  - Evaluation window
- Editing data
  - Right click on the variable name in the evaluation window
  - Then choose “Edit value” or “Edit type”

CENTER FOR COMPUTATION & TECHNOLOGY

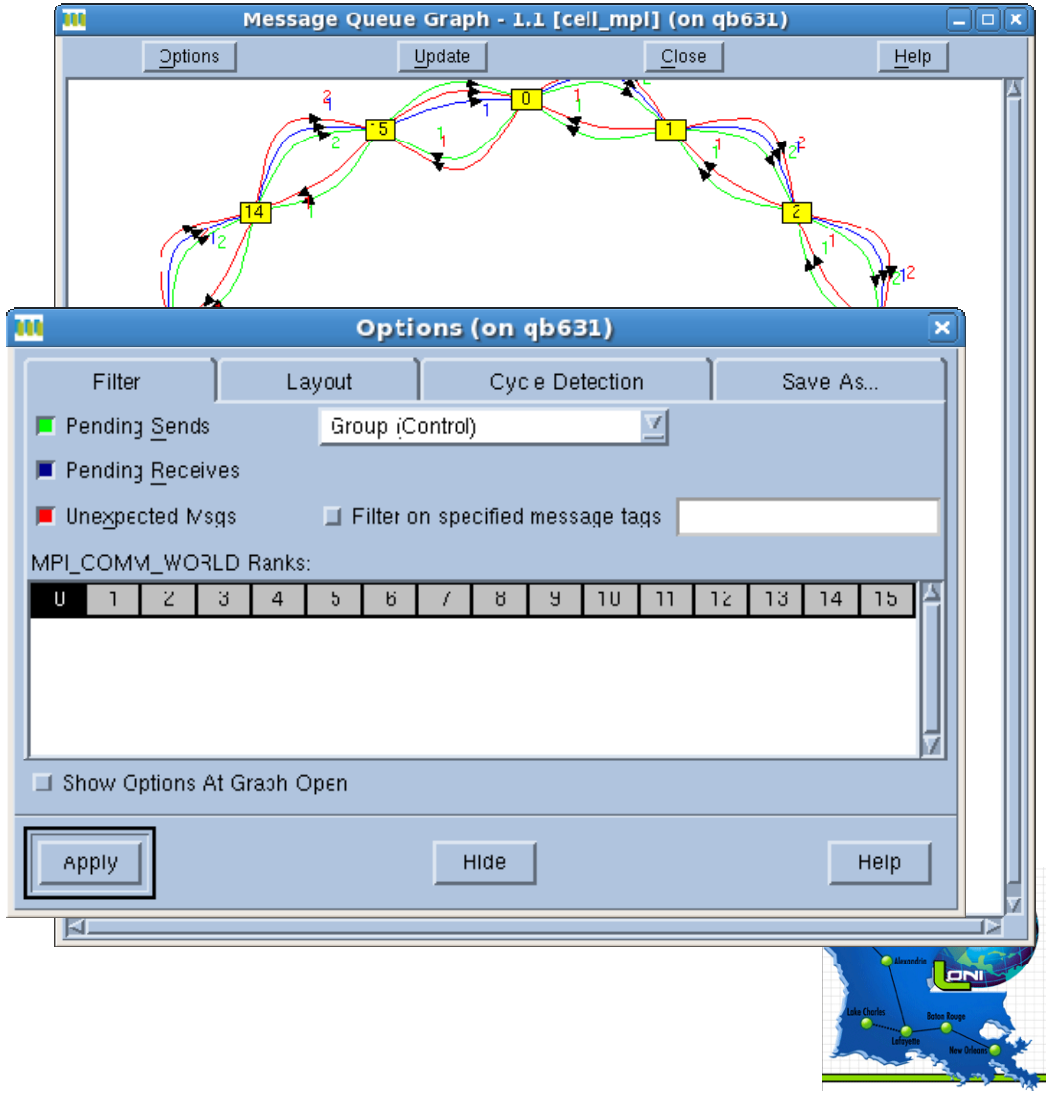
Viewing Dynamic Arrays in C/C++

- TotalView
  - Edit “type” in the variable window
  - Tell TotalView how to interpret the memory from a starting location
  - Example
    - To view an array of 100 integers
      - Int * -> int[100]*
- DDT
  - Drag a pointer variable into the evaluation window
  - Right click on the variable -> “View as vector”

CENTER FOR COMPUTATION & TECHNOLOGY

MPI Message Queues

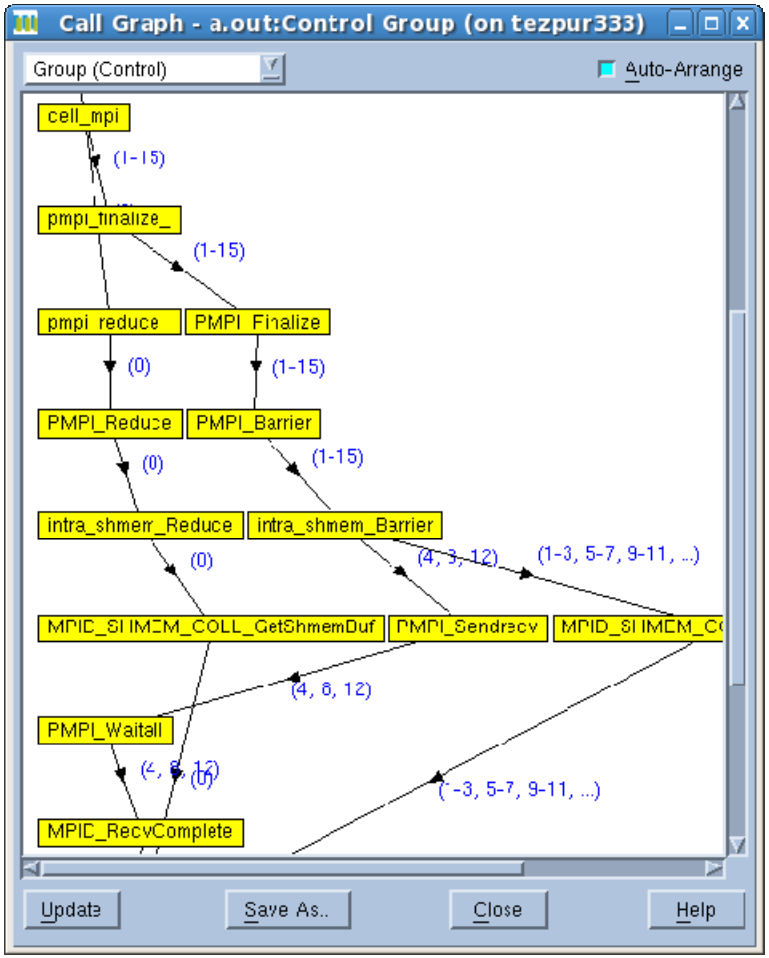
- Detect
  - Deadlocks
  - Load balancing issues
- TotalView
  - Tools -> Message Queue Graph
  - More options available
- DDT
  - View -> Message Queues



CENTER FOR COMPUTATION & TECHNOLOGY

TotalView - Call Graph

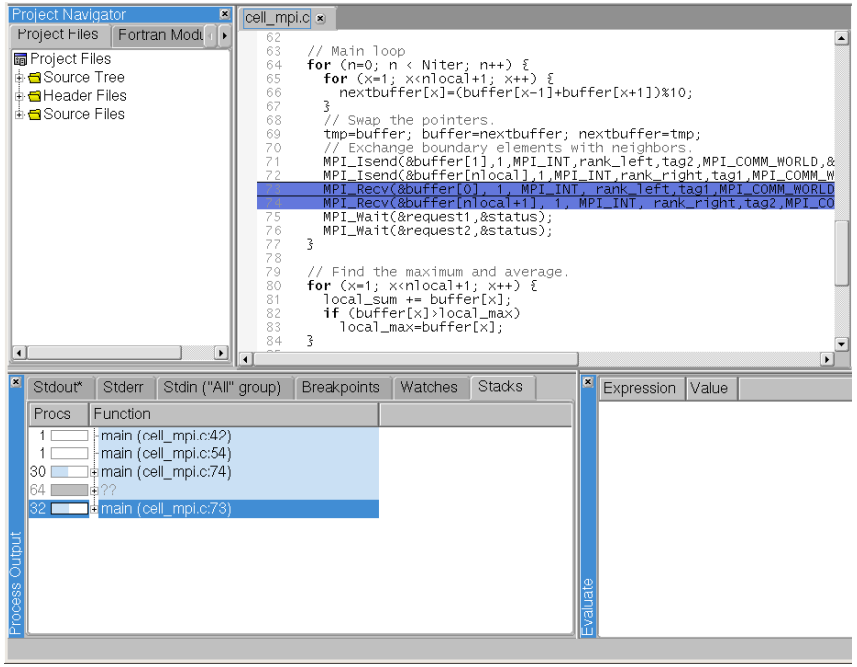
- Tools -> Call graph
- Quick view of program state
  - Nodes: functions
  - Edges: calls
- Look for outliers



CENTER FOR COMPUTATION & TECHNOLOGY

DDT - Parallel Stack View

- Allow users to see the position of each process/thread in the source code in same window
- Hover over any function to see a list of processes that are currently at that location



CENTER FOR COMPUTATION & TECHNOLOGY

References

- TotalView user manual
  - http://www.totalviewtech.com/support/documentation/totalview/index.html
- DDT user manual
  - http://www.allinea.com/downloads/userguide.pdf
- LLNL TotalView tutorial
  - https://computing.llnl.gov/tutorials/totalview
- NCSA Cyberinfrastructure Tutor
  - "Debugging Serial and Parallel Codes" course
- HPCBugBase
  - http://hpcbugbase.org/index.php/Main_Page